Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

SIMULATION VALUATION OF MULTIPLE EXERCISE OPTIONS

T. James Marshall

University of Western Ontario
1151 Richmond Street North
London, ON N6A5B7, CANADA

R. Mark Reesor

University of Western Ontario
1151 Richmond Street North
London, ON N6A5B7, CANADA

Matthew Cox

University of Western Ontario
1151 Richmond Street North
London, ON N6A5B7, CANADA

ABSTRACT

Multiple exercise options generalize American-style options as they allow the holder multiple exercise rights and control over the exercise amounts. They arise in both real and financial option applications, such as tolling agreements and swing options which are primarily used in the energy industry. The Forest of Stochastic Meshes is a recently proposed simulation method for valuing such options. This method accommodates general price processes and payoffs, produces high- and low-biased consistent estimators and a true option price confidence interval. Here we investigate improving the efficiency of this computationally-intensive procedure through high-performance computing (HPC) techniques.

1 INTRODUCTION

Multiple exercise right options may be considered as generalizations of American-style options as they provide the holder more than one exercise right. Examples of financial derivatives and real options with these properties have become more prevalent over the past decade and appear in sectors ranging from insurance to energy industries.

Tolling agreements are a real options example of multiple exercise options from the energy industry. Power markets have extremely high price volatilities making participants as well as potential lenders to producers wary of price risk. Other difficulties facing power producers include the facts that electricity is not a traded asset, generation costs depend on the price of the underlying physical asset(s), potential emissions costs, and power transmission to consumers. A tolling agreement is a structured transaction between a plant owner and typically a financial institution (FI). In the agreement, for an up front premium paid to the plant owner, the buyer is given the ability to determine when the plant is in operation and to take the electricity output at a predetermined price. Along with the operational constraints inherent to the plant the buyer is also often constrained by other contractual obligations. This agreement splits up the risks associated with power generation into operational risks associated with the plant and market risk associated with the fluctuating prices of the plant's output. It then leaves the management of these risk to the party involved which is best suited to handle them. The plant operator is guaranteed a stable cash flow from a FI with a typically higher credit rating which in turn makes it easier for the plant to raise capital to fund expansions and improvements, for example. The buyer of the contract uses the agreement to financially replicate the operation of a power plant without having to take on the associated operational risk.

In interest rate markets a chooser flexible cap is a multi-exercise interest rate derivative. These are a variant of an over the counter interest rate cap agreement. An interest rate cap is a derivative which consists of a series of caplets in which the i^{th} caplet provides the holder at time T_i with a payment of the notional multiplied by the difference in the current interest rate and a fixed strike rate, if positive, over N time intervals. These derivatives protect the holder, who may for example be an issuer of floating rate debt, from rises in the short term interest rates in exchange for an up front premium. A chooser flexible cap is almost identical to a interest rate cap except it only allows for $n < N$ caplets to be exercised at the holders discretion. The holder must decide how to optimally allocate the n exercise rights across the N opportunities. This multiple exercise option was valued in Meinshausen and Hambly (2004).

Another example is a swing option. Swing options have typically been used in energy markets to help producers manage the raw materials used in energy production in the face of uncertain demand. They allow the holder to have some control over both the timing and delivery amount of the underlying asset at predetermined prices. A swing option is typically noted as the swing portion of a base-loaded futures contract that gives a predetermined price for an amount of a commodity over a given period of time. The swing part of this overall contract allows for a variable delivery amount of the underlying above or below the amount determined by the base-loaded contract. However, the two pieces of the contract can be detached from one another and treated individually for valuation purposes. During the length of the contract the holder may exercise a given number swing rights. Typically these rights can only be exercised at a discrete set of times. Upon swinging, the holder can choose the additional volume bought/sold of the underlying asset. Refer to Jaillet, Ronn, and Tompaidis (2004) for a discussion on volume constraints and penalties.

As with the pricing of American-style options the valuation of multiple exercise options is a problem in stochastic optimal control. For American-style options the solution provides both a value and optimizing exercise rule, or stopping time. For multiple exercise options the solution also gives both a value and optimizing policy. In the case in which the holder controls only the exercise times (e.g., chooser flexible cap) the exercise policy is a sequence of stopping times. For options in which the holder controls the exercise times and amounts (e.g., swing options) the exercise policy is a paired sequence of stopping times and exercise amounts. The policy generalizes to other control variables.

In this work we focus on computational improvements to the Forest of Stochastic Meshes algorithm, a recently proposed method for valuing high-dimensional multiple exercise options (Marshall and Reesor 2011a). This is a discrete-time method in which the standard binomial or trinomial trees used to model the underlying price process in the Forest of Trees algorithm (Lari-Lavassani, Simchi, and Ware 2001) are replaced with stochastic meshes (Broadie and Glasserman 2004). High- and low-biased estimators are constructed which are consistent for the true option value. Furthermore the method can accommodate general price processes and payoffs and a confidence interval for the true value can easily be constructed. This algorithm is computationally intensive and we investigate the use of High Performance Computing (HPC) techniques to decrease the computing time. In particular, we focus on parallel processing implementations with central processing units (CPU) and general purpose graphics processing units (GPU), showing that substantial improvements can be realized.

The rest of this paper is organized as follows. The next section gives an overview of the approaches used for valuing multiple-exercise options, describes the stochastic mesh for valuing American-style options and then details the Forest of Stochastic Meshes algorithm. This is followed by a swing option pricing example that shows both the method's effectiveness and the computational resources required by a naive implementation without leveraging parallel processing. Our approaches on the use of HPC techniques to improve computational time are described and some numerical results on their effectiveness are presented in Section 4. Section 5 concludes the paper and outlines some directions for future work.

2 VALUATION OF MULTIPLE EXERCISE OPTIONS

Multiple exercise option-valuation algorithms are generalizations of those used for pricing American-style options, with most focused on swing options. A continuous-time approach is taken by Dahlgren and Korn (2005), Dahlgren (2005). For the time-discretized problem, the Forest of Trees (Lari-Lavassani, Simchi, and Ware 2001, Jaillet, Ronn, and Tompaidis 2004) is the simplest method. An analogous extension of partial differential equation methods appears in Kjaer (2008). Stochastic dynamic programming algorithms appear in Haarbrücker and Kuhn (2009) and Pflug and Broussev (2009). Simulation methods for multi-exercise option valuation include those that parameterize the set of exercise level curves (Ibáñez 1996) and those that use state space aggregation (Bally, Pagés, and Printems 2005). The Least-squares Monte Carlo and duality methods have been modified for the pricing of multi-exercise options in Barrera-Esteve, Bergeret, Dossal, Gobet, Meziou, Munos, and Reboul-Salze (2006) and Meinshausen and Hambly (2004), Aleksandrow and Hambly (2010), Bender (2011), respectively. A mathematical treatment of the discrete-time multiple stopping-time problem is given in Carmona and Touzi (2008) which also proposes a simulation-based valuation scheme using Malliavin calculus.

The Forest of Stochastic Trees (Marshall and Reesor 2011b) and Forest of Stochastic Meshes (Marshall and Reesor 2011a) are two recently proposed simulation based algorithms for valuing multiple exercise options. The Forest of Stochastic Trees replaces binomial with stochastic trees (Broadie and Glasserman 1997) in the Forest of Trees algorithm. Similarly, the Forest of Stochastic Meshes replaces binomial trees with stochastic meshes in the Forest of Trees. Since we focus on computational improvements for the Forest of Stochastic Meshes we provide a detailed description of the stochastic mesh and the Forest of Stochastic Meshes.

2.1 Stochastic Mesh

Assume the underlying stochastic variables $\{X\}$ are generated from a discrete time Markov Chain and are defined on a standard risk-neutral filtered and complete probability space $(\Omega, \mathscr{F}, \mathbb{Q})$ where Ω is the set of possible outcomes, $\mathscr{F}$ is a σ-Algebra, and $\mathbb{Q}$ is a risk-neutral probability measure. Let $\{\mathscr{F}_t\}$ be the filtration generated by $\{X_t\}$ (in general we only require that $\{X_t\}$ is adapted to $\{\mathscr{F}_t\}$) and $\{0,1,2,\ldots,m\}$ be the set of discrete times. Valuation of an American-style option via discrete-time dynamic programming is accomplished using the recursive equations

$$\begin{aligned} g_t &= \mathbb{E}_t[V_{t+1}] \quad \text{and} \\ V_t &= \max\{h_t(X_t), g_t\}, \end{aligned} \tag{1}$$

for $t = 0,\ldots,m-1$, where X_t, h_t, g_t and V_t are the time-t state variables, exercise, continuation, and option values, respectively and $\mathbb{E}_t[\cdot]$ denotes time-t conditional expectation. The terminal condition, $V_m = h_m(X_m)$, completes the dynamic program. Note that without loss of generality and with a gain in clarity we omit the discount factor in the valuation equations.

The stochastic mesh (Broadie and Glasserman 2004) is an effective simulation-based method for valuing American-style options. Construction of the mesh begins with the initial state vector X_0 and then the state transition function is applied to produce b independent state vector paths X_t^i for $i = 1,\ldots,b$ and $t = 1,\ldots,m$, where b is the chosen mesh size. This is illustrated on the left panel of Figure 2.1 with $b = 3$ and $m = 3$. The mesh is constructed from the simulated paths by connecting all time-t states, X_t^i, $i = 1,\ldots,b$ with all time-$(t+1)$ states, X_{t+1}^i, $i = 1,\ldots,b$. This is illustrated in the right panel of Figure 2.1. Suitably defined mesh weights, $\omega_t^{i,j}$, are used to describe transition probabilities between X_t^i and X_{t+1}^j (see (3) below).

An estimate to the value of the option, $\hat{V}_0$, is then determined by starting at option expiry and moving backward in time through the mesh to time 0; finding an approximate solution to (1). The recursive

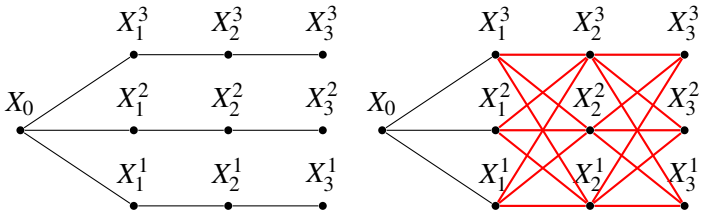

Figure 1: Stochastic Mesh with $m = 3$ timesteps and mesh size $b = 3$.

equations for value estimation are,

$$\begin{aligned} \hat{V}_m^i &= h_m(X_m^i), \\ \hat{g}_t^i &= \tfrac{1}{b}\textstyle\sum_{j=1}^b \omega_t^{i,j}\hat{V}_{t+1}^j, \quad \text{and} \\ \hat{V}_t^i &= \max\left(h_t(X_t^i), \hat{g}_t^i\right), \end{aligned} \tag{2}$$

for $t = 0,\ldots,m-1$. This replaces $g_t = \mathbb{E}_t\left[V_{t+1}\right]$ in (1) with the estimator $\hat{g}_t^i$.

To compute the time-t, state-i hold value estimator, $\hat{g}_t^i$, the time-$(t+1)$ option value estimators for all b states $(X_{t+1}^j, j = 1,\ldots,b)$ are used. These states are each generated from a different time-t state, with only one generated from X_t^i. Hence, one must adjust the transition probability to accommodate the fact that given X_t^i, the generated time-$(t+1)$ states are not independent and identically distributed. This accommodation is provided by the mesh weights in (3) and gives rise to the weighted average in (2) for $\hat{g}_t^i$.

As shown in Section 3 of Broadie and Glasserman (2004) an optimal choice for the mesh density functions leads to the weight functions,

$$\omega_t^{i,j} = \begin{cases} 1 & \text{for } t = 0 \\ \frac{f(t,X_t^i,X_{t+1}^j)}{b^{-1}\sum_{k=1}^b f(t,X_t^k,X_{t+1}^j)} & \text{for } t = 1,\ldots,m-1, \end{cases} \tag{3}$$

where $f(t,X_t^k,X_{t+1}^j)$ is the transition density associated with the underlying process.

With this we now have a complete description of the algorithm for the mesh estimator. In terms of computational intensity the effort involved in generating the state vectors is proportional to $b \times m$. The effort for evaluating the mesh via (2) is $b^2 \times m$. Hence the algorithm is quadratic in the mesh parameter b and linear in the number of exercise opportunities. Also note that the weighted average in (2) implies that a parallel implementation requires communication of the mesh weights between processors.

The estimator from (2) is high-biased and consistent. With a slight modification to (2) a low-biased consistent estimator, called a path estimator, can be generated. The path estimator is constructed by simulating a path, S, which is independent of the simulated mesh points. Along S the optimal exercise policy chooses to exercise the option at time $\tau^* = \min\{t : h_t(S_t) \geq g_t\}$ for a payoff of $h_{\tau^*}(S_{\tau^*})$. Replacing this optimal exercise policy with an approximate exercise policy given by $\hat{\tau}^* = \min\{t : h_t(S_t) \geq \hat{g}_t\}$ where $\hat{g}_t$ is an estimate of the time-t continuation value along path S, yields a suboptimal strategy, hence an underestimate of the option value.

The path estimator is computed from the following dynamic programming scheme,

$$\begin{aligned} \hat{P}_m^i &= h_m(S_m^i), \\ \hat{g}_t^i &= \tfrac{1}{b}\textstyle\sum_{j=1}^b \omega_t^{i,j}\hat{V}_{t+1}^j \quad \text{and,} \\ \hat{P}_t^i &= \begin{cases} \hat{P}_{t+1}^i & \text{if } h_t(S_t^i) \geq \hat{g}_t^i \\ \hat{g}_t^i & \text{if } h_t(S_t^i) < \hat{g}_t^i \end{cases}, \end{aligned} \tag{4}$$

for $t = 0, \ldots, m-1$ and for $i = 1, \ldots, n_p$, where n_p is the number of independent paths. Here $\hat{V}_t^i$ is the mesh estimator described in (2) with $\omega_t^{i,j}$ as in (3) and X_{t+1}^j, $j = 1, \ldots, b$ being the set of mesh points at time $t+1$. The computational effort for the path estimator is $n_p \times m$ to generate the paths and $n_p \times b^2 \times m$, where n_p is the number of independent paths, to evaluate the dynamic programming scheme in (4). This effort is in addition to the computational effort to evaluate the mesh estimator.

2.2 Forest of Stochastic Meshes

Consider a contract signed at time 0 that is in effect for time $\tau \in [T_1, T_2]$, where $0 \leq T_1 < T_2$. During the contract the holder may exercise a given number of rights. Typically these rights can only be exercised at a discrete set of times $\{\tau_1, \ldots, \tau_m\}$ with $T_1 \leq \tau_1 < \ldots < \tau_m \leq T_2$. Without loss of generality we take $\tau_m = T_2$. Along with exercise rights some contracts provide the holder the ability to control some other feature such as the volume bought/sold of the underlying asset. Here we refer to this other control as volume or usage, though it could refer to something else.

The valuation of multiple exercise options is a stochastic optimal control problem with three relevant state variables; underlying state variable (X), number of exercise rights remaining ($\mathscr{N}$), and the usage level (U) assuming some volume control. At each exercise opportunity and given $(X, \mathscr{N}, U)$, the current values of the state variables, the holder must choose between

- exercising u units plus continuing with an option having $\mathscr{N} - 1$ remaining exercise rights and usage level $U + u$; and
- continuing with an option having $\mathscr{N}$ exercise rights and usage level U (i.e., no exercise).

Note that with volume control the payoff from exercising u units changes with u (as does the continuation value of the option). Thus, the holder chooses the value-maximizing u when deciding to exercise.

Dynamic programming can be used for valuing multiple exercise options. In all variables, let the subscript t denote time and let $\mathscr{U}_t$ be the time-t set of admissible volume choices. The recursive equations for the dynamic program are

$$\begin{aligned} g_t^{\mathscr{N}_{t+1},U_{t+1}}(X_t) &= \mathbb{E}[V_{t+1}^{\mathscr{N}_{t+1},U_{t+1}}(X_{t+1})|X_t, \mathscr{N}_{t+1}, U_{t+1}], \quad \text{and} \\ V_t^{\mathscr{N}_t,U_t}(X_t) &= \max\left(\sup_{u\in\mathscr{U}_t}\left[h_t(X_t, \mathscr{N}_t, U_t, u) + g_t^{\mathscr{N}_t-1,U_t+u}(X_t)\right], g_t^{\mathscr{N}_t,U_t}(X_t)\right), \end{aligned} \tag{5}$$

with the terminal conditions

$$\begin{aligned} V_m^{\mathscr{N}_m,U_m}(X_m) &= \max\left(\sup_{u\in\mathscr{U}_m}\left[h_m(X_m, \mathscr{N}_m, U_m, u) + g_m^{\mathscr{N}_m-1,U_m+u}(X_m)\right], g_m^{\mathscr{N}_m,U_m}(X_m)\right), \quad \text{and} \\ g_m^{\mathscr{N}_m,U_m}(X_m) &= 0, \end{aligned} \tag{6}$$

where $g_t^{\mathscr{N},U}(X)$ and $V_t^{\mathscr{N},U}(X)$ are the time-t, state-$(X, \mathscr{N}, U)$ continuation and option values, respectively and $h_t(X, \mathscr{N}, U, u)$ is the payoff from exercising u units. This dynamic program can be modified in obvious ways to accommodate different multiple exercise option specifications. For example a swing option contract may specify a certain number of *up* and *down* swing rights, $\mathscr{N}_u$ and $\mathscr{N}_d$. Another variation is to allow for multiple rights to be exercised at each opportunity where each right corresponds to a fixed volume amount (Aleksandrow and Hambly 2010, Bender 2011).

The Forest of Trees method generalizes the tree method for pricing American options. In this method, many replications of the tree describing the underlying state variable process are used, each corresponding to a different state (number of rights remaining, usage level). Only a discrete finite set of volume choices is allowed, otherwise there would be an infinite number of trees in the forest. The dynamic program in (5) and (6) is solved exactly using the trees to compute the continuation value for each state.

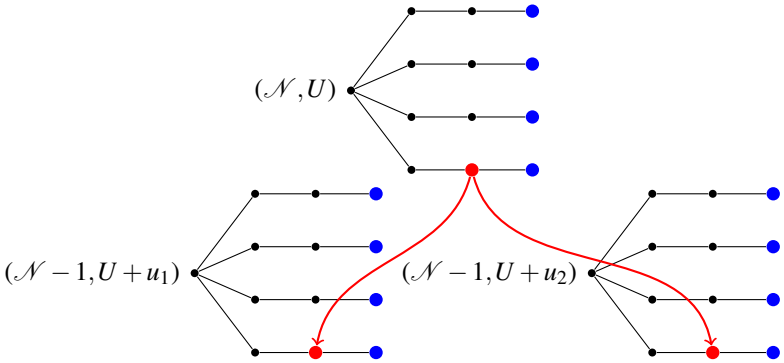

Figure 2: Section of a Forest of Meshes, $\mathscr{N}$ = # of exercise rights remaining, U = usage level.

In the Forest of Stochastic Meshes, the continuation values in (5) and (6) are replaced by stochastic mesh-type estimators. As with the original stochastic mesh technique, high- and low-biased option value estimators are constructed by using the analogous mesh and path continuation value estimators, respectively, on each mesh in the forest. The recursive equations for the high-biased mesh estimator are

$$\begin{aligned}
\hat{g}_t^{\mathscr{N}_{t+1},U_{t+1}}(X_t^i) &= \frac{1}{b}\sum_{j=1}^{b}\omega_t^{i,j}\hat{V}_{t+1}^{\mathscr{N}_{t+1},U_{t+1}}(X_{t+1}^j), \quad \text{and} \\
\hat{V}_t^{\mathscr{N}_t,U_t}(X_t^i) &= \max\left(\max_{u\in\mathscr{U}_t}\left[h_t(X_t^i,\mathscr{N}_t,U_t,u)+\hat{g}_t^{\mathscr{N}_t-1,U_t+u}(X_t^i)\right],\hat{g}_t^{\mathscr{N}_t,U_t}(X_t^i)\right),
\end{aligned} \tag{7}$$

with the terminal conditions

$$\begin{aligned}
\hat{V}_m^{\mathscr{N}_m,U_m}(X_m^i) &= \max\left(\max_{u\in\mathscr{U}_m}\left[h_m(X_m^i,\mathscr{N}_m,U_m,u)+\hat{g}_m^{\mathscr{N}_m-1,U_m+u}(X_m^i)\right],\hat{g}_m^{\mathscr{N}_m,U_m}(X_m^i)\right), \quad \text{and} \\
\hat{g}_m^{\mathscr{N}_m,U_m}(X_m^i) &= 0,
\end{aligned} \tag{8}$$

where $\hat{g}_t^{\mathscr{N},U}(X)$ and $\hat{V}_t^{\mathscr{N},U}(X)$ are the time-t, state-$(X,\mathscr{N},U)$ continuation and option values estimators, respectively and $h_t(X,\mathscr{N},U,u)$ is the time-t, state-$(X,\mathscr{N},U)$ payoff from exercising u units, b is the mesh size and $\omega_t^{i,j}$ are the mesh weights as given in (3).

Figure 2 is a diagram of a section in a forest of meshes with two volume choices, u_1 and u_2. It illustrates the nodes in the forest which need be considered when making an exercise decision given state $(\mathscr{N},U)$. The three choices are no exercise, exercise u_1 units, and exercise u_2 units. The red nodes are the equivalent nodes on the different meshes and the blue nodes are those which are used to calculate the continuation value estimators for the different exercise choices.

The path estimator is similarly defined using the path estimator on each mesh.

$$\begin{aligned}
\hat{g}_t^{\mathscr{N}_{t+1},U_{t+1}}(X_t^i) &= \frac{1}{b}\sum_{j=1}^{b}\omega_t^{i,j}\hat{V}_{t+1}^{\mathscr{N}_{t+1},U_{t+1}}(X_{t+1}^j), \\
\hat{H}_t^{\mathscr{N}_t,U_t}(S_t^i) &= \max\left(\max_{u\in\mathscr{U}_t}\left[h_t(S_t^i,\mathscr{N}_t,U_t,u)+\hat{g}_t^{\mathscr{N}_t-1,U_t+u}(X_t^i)\right],\hat{g}_t^{\mathscr{N}_t,U_t}(X_t^i)\right), \quad \text{and} \\
\hat{P}_t^{\mathscr{N}_t,U_t}(S_t^i) &= h_t(S_t^i,\mathscr{N}_t,U_t,\hat{u}^*)+\hat{P}_{t+1}^{\mathscr{N}_t-I,U_t+\hat{u}^*}(S_{t+1}^i)
\end{aligned} \tag{9}$$

where $\hat{H}_t^{\mathscr{N}_t,U_t}(S_t^i)$ is the continuation value, $\hat{u}^*$ is the estimated optimal exercise amount ($\hat{u}^* \geq 0$) and I is 0 if $\hat{u}^* = 0$ and 1 otherwise. The terminal conditions associated with this dynamic programming scheme are,

$$\hat{P}_m^{\mathscr{N}_m,U_m}(S_m^i) = \max\left(\max_{u\in\mathscr{U}_m}\left[h_m(S_m^i,\mathscr{N}_m,U_m,u) + \hat{g}_m^{\mathscr{N}_m-1,U_m+u}(X_m^i)\right], \hat{g}_m^{\mathscr{N}_m,U_m}(X_m^i)\right), \quad \text{and}$$
$$\hat{g}_m^{\mathscr{N}_m,U_m}(X_m^i) = 0. \tag{10}$$

Properties of the Forest of Stochastic Meshes estimators are inherited directly from the original mesh estimators. That is, the estimators are biased (high/low) and are consistent. Averaging the mesh and path estimators across independent repeated valuations results in high- and low-biased estimators having smaller standard error (than a single valuation) and allows one to construct confidence intervals for each. Taking the upper and lower confidence limits for the mesh and path estimators, respectively, results in a confidence interval for the true option value.

3 NUMERICAL EXAMPLE

Here we illustrate the Forest of Stochastic Meshes by pricing a swing option whose value depends on the price evolution of five stocks. This example is an extended version of one appearing in Broadie and Glasserman (1997) and Broadie and Glasserman (2004) illustrating the stochastic tree and mesh techniques for valuing American-style options. The method of mesh construction is that given above with the number of independent paths equaling the mesh size. Each underlying asset (stock) price, S^k, follows a risk–neutralized geometric Brownian motion (GBM) described by

$$dS_t^k = S_t^k\left[\left(r-\delta^k\right)dt + \sigma^k dZ_t^k\right], \tag{11}$$

where r is the risk-free rate of interest, δ^k is the continuously-paid dividend rate, σ^k is the volatility and Z^k is a standard Brownian motion. Furthermore, we assume that stock prices evolve independently and that r, δ^k and σ^k are constants. Finally, we assume that all stocks have the same initial value, S_0.

Upon exercising u units, the payoff of the option is

$$u \times \left(\max(S^1,S^2,\ldots,S^5) - K_u, K_d - \max(S^1,S^2,\ldots,S^5)\right)^+, \tag{12}$$

where K_u and K_d are the up and down strike prices respectively. We distinguish between up and down swing rights with $\mathscr{N}_u$ and $\mathscr{N}_d$ denoting the number of such rights, respectively.

To completely specify the parameters of this example, we take an option expiry of $T = 1$ year, $r = 0.05$, $\delta^k = 0.1$ and $\sigma^k = 0.2$ for $k = 1,\ldots,5$. In addition $K_u = K_d = 40$ there are no penalties and the volume exercised is held constant at 1 (i.e., take $u = 1$ in (12) and the holder has no control over the amount exercised). The number of up and down swing rights are $\mathscr{N}_u = \mathscr{N}_d = 2$ and the number of exercise opportunities is 6.

Using the above parameters the option is valued for different mesh size and number of repeated valuation pairs, (b,R), with changes to R satisfying $R = 16384(\frac{1000}{b})$. These are given in Table 1 along with the approximate total CPU time. It is worth noting that the number of repeated valuations is chosen to produce estimator standard errors of approximately 0.01%. Furthermore, increasing R has no effect on estimator bias. Only by increasing the mesh size can estimator bias be reduced with this algorithm.

Figure 3 plots the mesh and path option value estimates as a function of the mesh size. The high-biased mesh estimator decreases with mesh size while the low-biased path estimator increases with mesh size. Both estimators appear to be converging towards the same value as the mesh size increases. Due to the scale, confidence intervals for each estimator are not noticeable in Figure 3. With a branching factor of 128,000 the confidence interval for the true price has a width of approximately \$0.18 which represents about 1.3% of the option value (approximately \$14.88). Improving this interval estimate of option value requires increasing the mesh size.

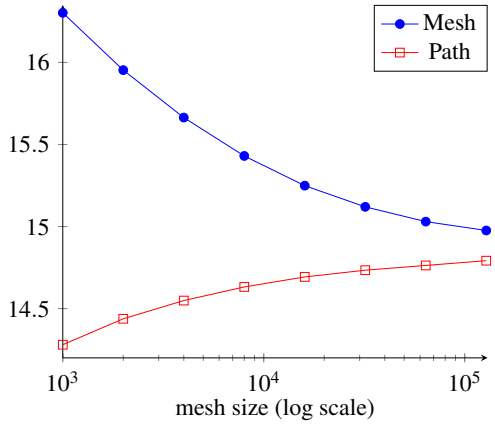

Figure 3: Mesh and path option-value estimates versus (log) mesh size.

All simulations for this example were completed on the SHARCNET cluster Hound. Hound consists of Opteron 2.2 GHz processors with 128 GB of memory. Run time is determined by 3 factors, the calculation time for a single forest, the number of repeated valuations desired and the number of processors available. The timing results given in the Serial time column of Table 1 are given in total CPU time across all valuations. Note that simultaneously doubling b and halving R results in a doubling of the computational time due to the quadratic scaling of computational effort with mesh size, implying a tradeoff between computational time (mesh size) and accuracy (estimator bias).

Table 1: Approximate total run time for various combinations of mesh size and number of repeated valuations (b,R). Serial refers to serial computing, MPI external refers to a naive parallel implementation (64 CPUs), MPI internal refers to an internally parallelized implementation (64 CPUs) with serial processing of the repeated valuations and GPGPU refers to a graphics processing unit implementation with serial processing of the repeated valuations.

(b,R)	Serial	MPI external	MPI internal	GPGPU
(4032,1)	105 sec	105 sec	1.9 sec	2.7 sec
(8000,1)	464 sec	464 sec	8.1 sec	10.7 sec
(16000,1)	40.78 min	40.78 min	0.64 min	0.70 min
(32000,1)	174.38 min	174.38 min	2.58 min	2.87 min
(4032,4096)	5 days	113 mins	132 mins	187 mins
(8000,2048)	11 days	248 mins	278 mins	366 mins
(16000,1024)	29 days	11 hrs	11 hrs	12.0 hrs
(32000,512)	62 days	23 hrs	22 hrs	24.5 hrs

4 COMPUTATIONAL IMPROVEMENTS THROUGH HPC

To reduce the run time for the Forest of Stochastic meshes we investigate HPC methods involving both CPUs and GPGPUs. Our first method of using HPC techniques to decrease the computational time involved a naive parallel implementation of the repeated valuations across many CPUs (i.e., essentially serial farming of the independent repeated valuations). No communication between CPUs is required to evaluate a given forest. Increasing the number of CPUs by a factor of n decreases the run time by a factor of $\frac{1}{n}$. Approximate timing results from this implementation can be found in the MPI external column of Table 1 and is computed by dividing the serial timing results by 64, the number of processors used. Note that for a single valuation

($b = 1$), the run time is the same as that for serial processing and if the number of repeated valuations is not a multiple of the number of processors then some CPUs will be idle (see right panel of Figure 4).

The second method involves only CPUs and we perform an internal parallelization of the meshes, which requires communication between CPUs, and perform the repeated valuations by serial processing. There are three major computational components of the simulation i) generation of the state; ii) computation of the weight denominators; and iii) calculations of weights, hold value, and exercise decision at each node in the mesh. In our scheme, CPUs are assigned tasks in the following manner

1. The underlying state vector mesh is generated completely and stored. To generate this mesh, the sample paths are generated in parallel with a single CPU generating an entire path from 0 to T. Upon path completion a CPU leapfrogs to the next sample path to be generated. For example if 256 CPUs are used the sample paths are generated in batches of 256, one for each available CPU.
2. As can be seen from Equation (3) all weights going into a given node share the same denominator. There are $b(m-2)$ of these denominators (where m is the number of time steps) as compared to b^2m weights. As such the denominators are calculated and stored prior to the evaluation of the meshes. Here a single processor is given the entire calculation of an individual denominator. Once this is complete the processor leapfrogs to the next available denominator calculation. Calculations performed in this way effectively reduce the computing work for the weights by a factor of $\frac{1}{b}$.
3. Beginning at expiry the meshes are evaluated. At a given time-step all work to be done for the ith node on all meshes in the forest is given to a single processor and then that processor leapfrogs to the next available set of nodes. That is the loop that is parallelized is the loop over sample paths for the high-biased mesh estimator. The loop over the forest index (indicating which mesh in the forest) is internal to the loop over sample paths for the mesh estimator. The work at each node entails: i) computing the weights needed for the hold value calculation using the stored denominators (not done at expiry); ii) the hold value calculation (not done at expiry); and iii) the exercise decision.
4. The results for a given time step are then collected on all processors using an MPI AllReduce before moving back to the prior time step and repeating.

The left panel of Figure 4 plots the run time (normalized to the run time of a single CPU) against the number of CPUs for the internal parallelization scheme described above and with $(b,R) = (2000,1024)$. There is a near perfect tradeoff between run time and number of processors. That is, increasing the number of processors by a factor of n reduces the run time by a factor of $\frac{1}{n}$. This is similar to the efficiency gains realized by the naive parallel implementation. The reason for this is that the communication time is negligible compared to the overall run time (see Table 2). The communication time given in Table 2 is the total (across all CPUs) average waiting time until other CPUs that are working on shared information are finished. This is a blocking communication protocol. Note that although the run time monotonically decreases with the number of processors, the communication time does not. This is due to delays from system synchronization. The run time for this implementation is given in the MPI internal column of Table 1. For a single valuation there is a significant reduction in the run time over serial computing and the naive parallel implementation since all processors are utilized. For example, the internal parallel scheme can perform a single valuation with $b = 32000$ in approximately the same time as the naive parallel implementation takes for $b = 4032$, producing less biased estimators. The right panel of Figure 4 plots the run time versus number of repeated valuations for $b = 16000$ and for 64 CPUs showing that the internal parallelization is better than the naive implementation over that range of R (though as R increases its advantage over the naive parallel implementation diminishes due to the communication time).

In our third method, a FirePro V9800 graphics accelerator with 1600 processing elements is used for valuation, with repeated valuations done in serial. The shared memory on the GPU allows for communication between processing units, effectively results in an internal parallelization of the meshes and is implemented by assigning parallel tasks using the OpenCL framework (an open specification application programming

Table 2: Total run time and communication time versus number of processors for an internal CPU parallelization of the meshes with $(b,R) = (2000, 1024)$.

# procs	run time (sec)	avg comtime (sec)
2	13875.6	51.7
4	6962.7	58.0
8	3608.0	72.4
16	1858.9	91.0
20	1504.3	128.1
40	798.4	84.9

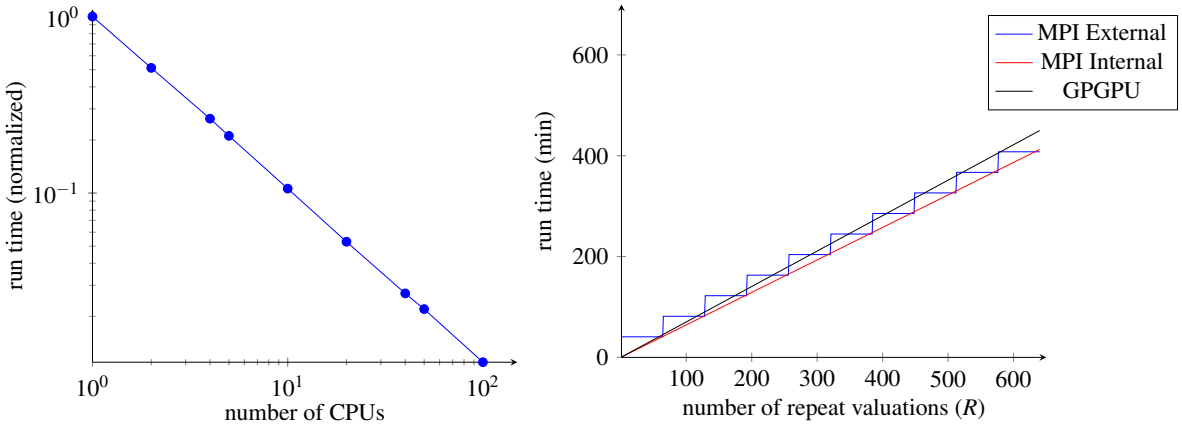

Figure 4: Left panel is the run time (normalized to the run time of a single CPU) versus number of CPUs for an internal CPU parallelization of the meshes with $(b,R) = (2000, 2014)$. Right panel is the run time versus the number of repeated valuations (R) for mesh size $b = 16000$. CPU implementations use 64 processors.

interface) from a single CPU as described below. Note that many modifications to the code (not detailed here due to space constraints) have resulted in a highly efficient GPGPU implementation.

1. The state vector mesh is generated by a single OpenCL kernel call. The approach in Phillips, Anderson, and Glotzer (2011) is used to generate the random numbers used for the paths, with care taken to draw unique numbers across repeated valuations. On GPUs, this method is much more efficient than other traditional methods (e.g., Mersenne Twister).
2. Prior to the evaluation of the meshes the mesh weight denominators (Equation (3)) are all calculated using an OpenCL kernel. Exploiting the memory hierarchy of the GPU is crucial to the performance of this phase of computation, particularly by holding batches of global values. Substantial effort has gone into optimizing how this kernel assign tasks to compute units and exploiting the particular characteristics of the memory hierarchy, particularly by performing the memory-intensive summation using fast local memory.
3. To evaluate the forest at a particular time step, a hold value kernel is enqueued with three indices, one each for sample path, estimator type (path/mesh) and mesh. Once the hold values are computed, an exercise decision kernel compares the hold value on the current mesh with option value on all other accessible meshes. These two kernels use an in-place update of a global buffer, requiring concurrent storage only of two time steps of option values, thus optimizing memory usage. Another kernel is used to process the expiry time step where all hold values are 0.

The run time for this implementation is given in the GPGPU column of Table 1, showing that in terms of computing time the GPGPU implementation is similar to the internal parallel method. Hence the above comments about the internal parallel method regarding the bias of an estimator in a fixed amount of time (e.g., a single valuation) and the right panel of Figure 4 also apply. Additionally note that

1. We have shown that this valuation problem is well-suited for a GPU implementation as the computing time scales approximately with that of a naive parallel method using no communication between processors.
2. A single GPU costs a few hundred dollars, orders of magnitude less than the cost of CPU clusters, giving similar computational improvements for a much lower dollar cost.
3. Multiple GPUs can be used to perform the repeated valuations in parallel, reducing the computing time by the reciprocal of the number of GPUs.

5 DISCUSSION

We have shown that HPC techniques can significantly reduce the run times of the computationally intensive Forest of Stochastic Meshes. The naive parallel, internal parallel and GPGPU implementations are quite effective and all give approximately the same run time when performing repeated valuations that are a multiple of the number of CPUs used. For a single valuation, the internal parallel and GPGPU methods give similar computing times and are much faster than the naive parallel and serial methods. The low cost and ability to use multiple GPUs suggests this as the method of choice. Note that additional computational efficiencies can be realized through variance and bias-reduction techniques. The methods presented here can be combined with these to yield extremely efficient pricing algorithms for multiple exercise options.

ACKNOWLEDGMENTS

We thank Western, NSERC, MITACS, Bank of Montreal Capital Markets, and Alberta Treasury Branches for financial support and SHARCNET for computational resources. We thank Matt Davison, Rogemar Mamon, Adam Metzler, the Financial Mathematics Research Group, Piers Lawrence and Allan MacIsaac all from UWO. We also thank Tyson Whitehead and Baolai Ge from SHARCNET. All remaining errors are our responsibility.

REFERENCES

Aleksandrow, N., and B. M. Hambly. 2010. "A dual approach to multiple exercise option problems under constraints". *Mathematical Methods of Operations Research* 71:503–533.

Bally, V., G. Pagés, and J. Printems. 2005. "A Quantization Tree Method for Pricing and Hedging Multidimensional American Options". *Mathematical Finance* 15:119–168.

Barrera-Esteve, C., F. Bergeret, C. Dossal, E. Gobet, A. Meziou, R. Munos, and D. Reboul-Salze. 2006. "Numerical Methods for the Pricing of Swing Options: A Stochastic Control Approach". *Methodology and Computing in Applied Probability* 8:517–540.

Bender, C. 2011. "Dual pricing of multi-exercise options under volume constraints". *Finance and Stochastics* 15 (1): 1–26.

Broadie, M., and P. Glasserman. 1997. "Pricing American-style securities using simulation". *Journal of Economic Dynamics and Control* 21:1323–1352.

Broadie, M., and P. Glasserman. 2004. "A stochastic mesh method for pricing high-dimensional American options". *The Journal of Computational Finance* 7:35–72.

Carmona, R., and N. Touzi. 2008. "Optimal Multiple Stopping and Valuation of Swing Options". *Mathematical Finance* 18:239–268.

Dahlgren, M. 2005. "A Continuous Time Model to Price Commodity-Based Swing Options". *Review of Derivatives Research* 8:27–47.

Dahlgren, M., and R. Korn. 2005. "The Swing Option on the Stock Market". *The International Journal of Theoretical and Applied Finance* 8:123–129.

Haarbrücker, G., and D. Kuhn. 2009. "Valuation of Electricity Swing Options by Multistage Stochastic Programming". *Automatica* 45:889–899.

Ibáñez, A. 1996. "Valuation by Simulation of Contingent Claims with Multiple Early Exercise Opportunities". *Mathematical Finance* 19:19–30.

Jaillet, P., E. I. Ronn, and S. Tompaidis. 2004. "Valuation of Commodity-Based Swing Options". *Management Science* 50:909–921.

Kjaer, M. 2008. "Pricing of Swing Options in a Mean Reverting Model with Jumps". *Applied Mathematical Finance* 15:479–502.

Lari-Lavassani, A., M. Simchi, and A. Ware. 2001. "A Discrete Valuation of Swing Options". *Canadian Applied Mathematics Quarterly* 9:35–73.

Marshall, T. J., and R. M. Reesor. 2011a. "Forest of Stochastic Meshes: A New Method for Valuing High Dimensional Swing Options". *Operations Research Letters* 39 (1): 17–21.

Marshall, T. J., and R. M. Reesor. 2011b. "Forest of Stochastic Trees: A New Method for Valuing High Dimensional Swing Options". Technical report, Working Paper, Department of Applied Mathematics, The University of Western Ontario, London, Ontario, Canada.

Meinshausen, N., and B. M. Hambly. 2004. "Monte Carlo Methods For the Valuation of Multiple-Exercise Options". *Mathematical Finance* 14:557–583.

Pflug, G. C., and N. Broussev. 2009. "Electricity Swing Options: Behavioral Models and Pricing". *European Journal of Operations Reasearch* 197:1041–1050.

Phillips, C. L., J. A. Anderson, and S. C. Glotzer. 2011. "Pseudo-random number generation for Brownian Dynamics and Dissipative Particle Dynamics simulations on GPU devices". *Journal of Computational Physics* 0 (19): 7191–7201.

AUTHOR BIOGRAPHIES

T. JAMES MARSHALL is a doctoral candidate in the Department of Applied Mathematics at the University of Western Ontario. He received a B.Sc. in Mathematics and Physics from the University of New Brunswick and an M.Sc. in Applied Math (Theoretical Physics) from the University of Western Ontario. Prior to his doctoral studies he was a lecturer in the Department of Physics at the University of New Brunswick. His research interests are in computational quantitative finance. His email address is tmarsha8@uwo.ca.

R. MARK REESOR is an Associate Professor and SHARCNET Research Chair in Financial Mathematics in the Department of Applied Mathematics at the University of Western Ontario. Prior to joining UWO he was an analyst in the Financial Markets Department at the Bank of Canada. He received a B.Sc. in Mathematics from McGill University and a Masters in Mathematics (Statistics, Co-op) and Ph.D. (Statistics) from the University of Waterloo. He was a co-founder of the Committee-to-Establish the National Institute of Finance. His research interests include the application of stochastic models and simulation to problems in finance, insurance and risk management. His email is mreesor@uwo.ca.

MATTHEW COX is a doctoral candidate in the Applied Mathematics Department at the University of Western Ontario. He received a M.Math. from the University of Waterloo and a H.B.Sc. from the University of Western Ontario. His research interests include parallel processing for scientific computation in both physics-based and non-physical simulations, particularly the development of novel parallel implementations for GPU computation. His email address is mcox22@uwo.ca.